🏓 TANCAT SYSTEM - Guía Completa de Instalación

📋 Índice

🔧 Requisitos Previos

Software Necesario

- **Node.js** >= 16.0.0 (Descargar)
- **npm** >= 8.0.0 (incluido con Node.js)
- **PostgreSQL** >= 12.0 (Descargar)
- **Git** (Descargar)

Verificar Instalaciones

```bash
node --version    # Debe mostrar v16.0.0 o superior
npm --version     # Debe mostrar 8.0.0 o superior
psql --version    # Debe mostrar PostgreSQL 12 o superior
git --version     # Cualquier versión reciente
```

🖥️ Instalación del Backend

1. Navegar al Directorio del Backend

```bash
cd TANCAT-SYSTEM/backend
```

2. Crear archivo package.json

Crea el archivo `package.json` con el contenido proporcionado anteriormente.

3. Instalar Dependencias

bash

```
# Instalar todas las dependencias
npm install

# Verificar instalación
npm list --depth=0
```

4. Crear Estructura de Archivos

Crear archivo .env

bash

```
# Copiar el template
cp .env.example .env

# Editar con tus credenciales
nano .env
```

Configurar variables de entorno en .env:

bash

```
# Configuración del Servidor
NODE_ENV=development
PORT=3000
HOST=localhost

# Base de Datos PostgreSQL
DB_HOST=localhost
DB_PORT=5432
DB_NAME=tancat_db
DB_USER=postgres
DB_PASSWORD=tu_password_aqui

# JWT Secret (cambiar por una clave segura)
JWT_SECRET=tu_clave_super_secreta_aqui_cambiar_en_produccion
JWT_EXPIRE=24h

# CORS
CORS_ORIGIN=http://localhost:3000,http://127.0.0.1:3000,http://localhost:5500,http://127.0.0.1:5500
```

5. Crear Archivos del Backend

Crear los siguientes archivos en sus respectivas carpetas:

`server.js` (raíz del backend)

```
javascript

// Contenido del archivo server.js proporcionado anteriormente
```

`app.js` (raíz del backend)

```
javascript

// Contenido del archivo app.js proporcionado anteriormente
```

`utils/database.js`

```
javascript

// Contenido del archivo database.js proporcionado anteriormente
```

`controllers/clienteController.js`

```
javascript

// Contenido del clienteController.js proporcionado anteriormente
```

`routes/clientes.js`

```
javascript

// Contenido de las rutas de cliente proporcionadas anteriormente
```

`routes/auth.js`, `routes/admin.js`, etc.

```
javascript

// Contenido de las rutas adicionales proporcionadas anteriormente
```

🗄️ Configuración de Base de Datos

Opción A: PostgreSQL Local

1. Crear Base de Datos

```
bash
```

```
# Conectar a PostgreSQL
psql -U postgres

# Crear base de datos
CREATE DATABASE tancat_db;

# Crear usuario (opcional)
CREATE USER tancat_user WITH PASSWORD 'tu_password';
GRANT ALL PRIVILEGES ON DATABASE tancat_db TO tancat_user;

# Salir
\q
```

2. Ejecutar Script de Estructura

bash

```
# Conectar a la base de datos
psql -U postgres -d tancat_db

# Ejecutar el script SQL proporcionado
\i path/to/tancat_database_structure.sql
```

Opción B: Supabase (Recomendado para desarrollo)

1. Crear Proyecto en Supabase

1. Ir a supabase.com
2. Crear cuenta y nuevo proyecto
3. Esperar a que se complete la configuración

2. Obtener Credenciales

1. Ir a Settings > Database
2. Copiar: Host, Database name, Port, User, Password
3. Ir a Settings > API
4. Copiar: Project URL, anon public key, service_role key

3. Configurar .env para Supabase

bash

```
# Supabase
SUPABASE_URL=https://tu-proyecto.supabase.co
SUPABASE_ANON_KEY=tu_anon_key_aqui
SUPABASE_SERVICE_KEY=tu_service_key_aqui

# También mantener configuración PostgreSQL
DB_HOST=db.tu-proyecto.supabase.co
DB_PORT=5432
DB_NAME=postgres
DB_USER=postgres
DB_PASSWORD=tu_password_de_supabase
```

4. Ejecutar Estructura en Supabase

1. Ir al SQL Editor en Supabase
2. Copiar y ejecutar el script `tancat_database_structure.sql`

🌐 Configuración del Frontend

1. Navegar al Frontend

```bash
cd ../frontend
```

2. Configurar Archivos

Actualizar `assets/js/cliente-main.js`

```javascript
// Cambiar configuración para conectar al backend
const CONFIG = {
    API_BASE_URL: 'http://localhost:3000/api',
    MODO_OFFLINE: false  // Cambiar a false para usar backend real
};
```

3. Servir Frontend

Opción A: Live Server (VS Code)

1. Instalar extensión "Live Server"
2. Click derecho en `index.html`
3. Seleccionar "Open with Live Server"

Opción B: Servidor HTTP Simple

```bash
# Instalar servidor HTTP global
npm install -g http-server

# Ejecutar desde la carpeta frontend
http-server -p 5500 -c-1
```

Opción C: Python (si está instalado)

```bash
# Python 3
python -m http.server 5500

# Python 2
python -m SimpleHTTPServer 5500
```

🚀 Ejecución del Sistema

1. Terminal 1: Backend

```bash
cd TANCAT-SYSTEM/backend

# Desarrollo (con auto-reload)
npm run dev

# O producción
npm start
```

2. Terminal 2: Frontend

```bash
cd TANCAT-SYSTEM/frontend

# Si usas http-server
http-server -p 5500 -c-1

# O abrir en navegador directamente si usas Live Server
```

3. URLs de Acceso

- **Frontend**: http://localhost:5500
- **Backend API**: http://localhost:3000/api
- **API Docs**: http://localhost:3000/api/docs
- **Health Check**: http://localhost:3000/api/health

✅ Verificación

1. Verificar Backend

```bash
# Health check
curl http://localhost:3000/api/health

# Endpoint de sedes
curl http://localhost:3000/api/cliente/sedes

# Endpoint de deportes
curl http://localhost:3000/api/cliente/deportes
```

2. Verificar Frontend

1. Abrir http://localhost:5500
2. Verificar que no aparezca la barra naranja de "MODO DESARROLLO"
3. Probar selección de sede y deporte
4. Hacer una consulta de disponibilidad

3. Verificar Base de Datos

```bash
# Conectar a la base de datos
psql -U postgres -d tancat_db

# Verificar tablas
\dt

# Verificar datos de ejemplo
SELECT * FROM sedes;
SELECT * FROM deportes;
```

🔧 Solución de Problemas

Error: Puerto 3000 en uso

bash

```
# Encontrar proceso usando el puerto
lsof -i :3000

# Matar proceso (reemplazar PID)
kill -9 <PID>

# O cambiar puerto en .env
PORT=3001
```

Error: No se puede conectar a PostgreSQL

bash

```
# Verificar que PostgreSQL esté ejecutándose
sudo systemctl status postgresql  # Linux
brew services list | grep postgres  # macOS
```

Error: Módulos no encontrados

bash

```
# Reinstalar dependencias
rm -rf node_modules package-lock.json
npm install
```

Error: CORS

bash

```
# Verificar configuración CORS en .env
CORS_ORIGIN=http://localhost:5500,http://127.0.0.1:5500
```

Frontend muestra datos de ejemplo

javascript

```
// Verificar en cliente-main.js que esté configurado correctamente
const CONFIG = {
    MODO_OFFLINE: false  // Debe ser false
};
```

📦 Scripts NPM Disponibles

bash

```
# Desarrollo con auto-reload
npm run dev

# Producción
npm start

# Ejecutar migraciones
npm run migrate

# Ejecutar seeders
npm run seed

# Configurar base de datos completa
npm run db:setup

# Linting
npm run lint
npm run lint:fix

# Tests
npm test
```

🔄 Workflow de Desarrollo

1. Desarrollo Diario

bash

```
# Terminal 1: Backend
cd backend && npm run dev

# Terminal 2: Frontend
cd frontend && http-server -p 5500 -c-1
```

2. Cambios en Base de Datos

bash

```
# Aplicar cambios de estructura
npm run migrate

# Cargar datos de prueba
npm run seed
```

3. Testing

```bash
# API tests
curl http://localhost:3000/api/health
curl http://localhost:3000/api/cliente/sedes

# Frontend tests
# Abrir http://localhost:5500 y probar funcionalidad
```

📞 Soporte

Si encuentras problemas:

1. **Verificar logs**: Revisar la consola del backend y navegador
2. **Verificar configuración**: Asegurar que .env esté configurado correctamente
3. **Verificar puertos**: Asegurar que los puertos no estén en uso
4. **Verificar base de datos**: Asegurar conexión y estructura

Logs Importantes

- **Backend**: Consola donde ejecutaste `npm run dev`
- **Frontend**: Consola de desarrollo del navegador (F12)
- **Base de datos**: Logs de PostgreSQL

🎯 Próximos Pasos

Una vez que el sistema básico esté funcionando:

1. **Implementar autenticación** completa
2. **Agregar módulos de administración**
3. **Implementar gestión de reservas** real
4. **Agregar sistema de torneos**
5. **Implementar reportes** y analytics

¡El sistema base estará listo para desarrollo y expansión! 🚀